필드 위치결정을 위한 리팩토링 요인

정영애*, 박용범*
*단국대학교 전자계산학과
e-mail:{yajung;ybpark}@dankook.ac.kr

Refactoring factors to decide the location of a field

Young-Ae Jung*, Young B. Park*
*Dept of Computer Science, Dan-Kook University

요 약

소프트웨어는 생명주기 전반에 거쳐 발생하는 요구사항 변경으로 수정이 불가피하다. 소프트웨어를 수정할 때 품질과 안정성을 유지하는 것은 중요한 문제이다. 본 논문에서는 무브 메소드(Move Method) 기법을 기초로 하여 리팩토링 적용요인을 제안하고, 로지스틱 회귀분석을 통하여 적용요인이 메소드의 위치를 결정지을 수 있는 요인임을 증명하는 방법에 대하여 살펴본다. 또한 객체지향 프로그램에서 메소드와 더불어 중요한 요소인 필드의 위치를 결정지을 수 있는 요인을 무브 필드(Move Field) 기법에 기초하여, 객체내 필드의 위치를 결정하기 위한 리팩토링 요인을 정의하고, 향후 연구과제에 대하여 논한다.

1. 서론

소프트웨어는 생명주기 전반에 거쳐 끊임없이 변경되는 요구사항의 변경을 수용하기 위해 수정된다. 이런 소프트웨어 유지보수 활동은 프로그램 모듈검증과 변경사항에 대한 테스트가 소프트웨어를 사용하지 않을 때까지 지속적으로 이루어져야 한다는 점에서 상당한 비용이 요구된다[9].

소프트웨어 개발자들은 처음부터 완벽한 소프트웨어를 개발하고자 하지만 처음부터 완벽한 소프트웨어를 만드는 것은 불가능하다. 대부분의 경우 코드가 작성된 후 더 자주 수정된다. 유지보수단계에서 요구사항의 변경을 체계적이고 안전하게 수행하기 위한 방법의 하나로 리팩토링이 사용되어 왔다.

리팩토링은 대부분 전문가들의 경험에 의한 수동분석(manual analysis)을 통하여 이루어졌고 리팩토링의 체계적인 적용기준이 제공되지 못하였다[9].

하지만 최근에는 전문가의 경험을 바탕으로 리팩토링 작업의 상당부분을 자동화하려는 시도가 있었다[10,11].

리팩토링을 자동화하기 위해 필요한 객관적인 판단기준과 방법[4,5,9,10,11]도 다양하게 제시되어 왔다.

본 논문에서는 기존에 메소드 위치 결정 방법을 처리하기 위해 로지스틱 분석을 이용하여 그 결과를 추측할 수 있도록 한 연구방법[1]을 살펴보고, 그 연구방법에 기초하여 필드 위치 결정 요인을 정의하고자 한다.

메소드의 위치를 결정하기 위해 사용된 요인 분석 방법을 기초로 하여 필드의 위치를 결정지을 수 있는 요인을 리팩토링 기법 중의 하나인 무브 필드 기법의 적용여부를 결정하는 요인에서 추출하고, 정의하였다.

본 논문의 구성은 다음과 같다. 2장에서는 리팩토링과 자동화 동향을 살펴보고 메소드의 위치를 결정하기 위해 로지스틱 회귀분석을 사용한 사례를 살펴본다. 3장에서는 객체내 필드 위치결정을 위한 무브 필드 기법의 적용요인을 정의하고, 4장에서는 결론 및 향후 연구과제에 대하여 논한다.

2. 관련 연구

2.1 리팩토링 동향

리팩토링은 프로그램 행위(behavior)를 보전하면서 프로그램의 가독성, 구조, 성능, 유지보수성, 추상성 등을 향상시키는 좋은 방법이다[2]. 리팩토링은

1990년 초에 개발되었고[12,13,14], 그 후 소프트웨어 개발의 주된 분야가 되었다[8].

리팩토링 자동화는 리팩토링 후보영역을 정의하는 분야와 개발자 정의 후보영역에 리팩토링을 자동으로 적용하는 분야로 양분화 되었다[15]. 리팩토링 후보영역을 정의하는 분야로는 프로그램 불변점(program invariants) 패턴 매칭을 이용한 접근방법과 요구사항의 변화에 대하여 후보영역(candidate spot)을 추론규칙을 사용하여 원하는 디자인 패턴 구조로 변환하는 방법 등이 제안되었다[7,10].

개발자 정의 후보영역에 자동으로 리팩토링을 적용하는 영역에는, 프로그램 슬라이싱(Program slicing)을 사용하여 메소드를 자동으로 리팩토링하는 방식[6], 가중치 종속 그래프(weighted dependence graph)를 이용하여 객체지향 프레임워크에서 메소드를 자동으로 리팩토링하는 메커니즘[5]등이 제안되었다. 더불어 응집도 매트릭스(cohesion matrix)를 이용한 거리를 측정하여 시각화하는 툴을 개발하여 리팩토링을 돕는 연구도 제안된 바 있다[4].

2.2 로지스틱 회귀분석

로지스틱 회귀분석이란 독립변수의 양적인 변수를 이용해서 종속변수인 이변량적 변수들간의 인과관계를 추정하는 기법이며, 한 개의 종속 변수와 여러 개의 독립변수간의 상호관련성에 대해 분석하려 할 때 가장 널리 사용되는 통계기법이다[3].

종속변수의 척도가 연속형이 아니라 범주형으로 측정되기 때문에 객체 메소드나 필드의 위치분석에 로지스틱 회귀분석을 이용하는 것이 적합하다.

무브 필드의 적용요인으로 정의된 요인을 로지스틱 회귀분석의 독립변수로서 사용하여, 각 적용요인들과 무브 필드의 적용여부에 대한 인과관계를 추정하는 방법으로 사용될 것이다.

3. 필드위치결정을 위한 리팩토링 요인 정의

3.1 로지스틱분석을 이용한 메소드 위치결정방법

리팩토링의 기법 중 하나인 무브 메소드 기법 요인을 추출하여 그 요인을 메소드의 위치를 결정짓기 위한 요인으로 사용할 수 있는가를 로지스틱 분석을 통하여 검증하였다.

수동적 분석에 의해 리팩토링이 수행될 때 보편적으로 사용되는 요인들을 유도하였다. 다음은 무브 메소드의 적용요인을 유도하기 위해 명확한 용어를 정의한 것이다.

- 소스클래스(source class) : 무브 메소드를 적용할 메소드가 포함되어 있는 클래스
- 타겟클래스(target class) : 무브 메소드를 적용할 메소드가 이동하게 될 클래스
- 필드(field): 무브 메소드를 적용할 메소드가 참조하는 소스클래스의 필드 (타켓클래스의 필드 참조를 위한 get/set 메소드도 필드로 간주)
- 대상메소드: 무브메소드기법을 적용할 메소드
- 일반메소드(general method): 필드 참조를 위한 get/set 메소드를 제외한 모든 메소드

실험을 위하여 무브 메소드 리팩토링을 적용할 수 있는 프로그램을 선별하여 실험에 이용하였다. 각 프로그램별로 각 적용요인에 해당되어지는 개별적인 데이터를 구하였다. 각 적용요인에 의해 얻어진 데이터는 <표 1>과 같다.

소스클래스와 타겟클래스의 수는 각 적용요인에 따라 참조하는 필드개수나 참조횟수를 나타내며 수동분석결과는 실험에 사용된 프로그램의 프로그래머가 분석한 결과이다. <표 1>의 Factor 1-3의 값은 각 적용요인별 소스클래스와 타겟클래스의 값을 감산한 결과이다.

<표 1> 정의된 적용요인 데이터와 Factor 데이터

프로그램 번호	적용요인1		Factor1	적용요인2		Factor2	적용요인3		Factor3	수동분석결과
	소스클래스	타겟클래스		소스클래스	타겟클래스		소스클래스	타겟클래스		
1	0	0	0	0	1	-1	0	1	-1	1
2	1	0	1	1	1	0	0	1	-1	0
3	0	2	0	0	2	-2	1	0	1	1
4	0	0	0	1	0	1	0	0	0	0
5	0	3	0	1	0	1	1	0	1	1
6	2	0	1	0	0	0	1	0	1	1
7	2	0	1	0	0	0	1	0	1	1
8	2	0	1	0	0	0	1	0	1	1
9	2	0	1	0	0	0	1	0	1	1
10	1	2	0	0	0	0	1	0	1	1
11	2	0	1	0	0	0	1	0	1	1

각 적용요인의 조합에 따른 결과의 차이를 파악하기 위하여, 세 적용요인을 모두 적용한 경우, 세 가지 적용요인 중 두 가지씩 짝을 지은 세 가지의 경우, 개별적인 경우 세 가지에 대하여 분석을 수행하였다.

각 적용요인의 조합에 따른 결과의 변화를 파악하기 위하여, 세 가지 적용요인을 모두 적용한 경우, 세 가지 적용요인 중 두 가지씩 짝을 지은 3가지의 경우, 개별적인 경우 3가지에 대하여 분석을 수행하였다. 그 결과는 <표 2>와 같다.

세 가지 적용요인을 모두 적용한 경우와 적용요인

<표 2> 각 요인의 경우의 수에 따른 분석결과

프로그램 번호	수동 분석 결과	Factor 1,2,3		Factor 1,3		Fcator 1,2		Factor 2,3		Factor 1		Factor 2		Factor 3	
		예상 확률	예상 그룹	예상 확률	예상 그룹	예상 확률	예상 그룹	예상 확률	예상 그룹	예상 확률	예상 그룹	예상 확률	예상 그룹	예상 확률	예상 그룹
1	1	0.9999999966	1	0.3397018856	0	1.0000000000	1	0.9999999904	1	0.8000000000	1	0.9775400000	1	0.3120000000	0
2	0	0.0000000000	0	0.2706739793	0	0.8333333333	1	0.0000000071	0	0.8333333321	1	0.8650000000	1	0.3120000000	0
3	1	1.0000000000	1	0.9603499280	1	1.0000000000	1	1.0000000000	1	0.8000000000	1	0.9966300000	1	0.9530000000	1
4	0	0.0000000104	0	0.7792482632	1	0.5000000000	1	0.0000000061	0	0.8000000000	1	0.4854000000	0	0.7520000000	1
5	1	0.9999999894	1	0.9603499280	1	0.5000000000	1	0.9999999869	1	0.8000000000	1	0.4854000000	0	0.9530000000	1
6	1	0.9999999997	1	0.9458651665	1	0.8333333333	1	1.0000000000	1	0.8333333321	1	0.8650000000	1	0.9530000000	1
7	1	0.9999999997	1	0.9458651665	1	0.8333333333	1	1.0000000000	1	0.8333333321	1	0.8650000000	1	0.9530000000	1
8	1	0.9999999997	1	0.9458651665	1	0.8333333333	1	1.0000000000	1	0.8333333321	1	0.8650000000	1	0.9530000000	1
9	1	0.9999999997	1	0.9458651665	1	0.8333333333	1	1.0000000000	1	0.8333333321	1	0.8650000000	1	0.9530000000	1
10	1	1.0000000000	1	0.9603499280	1	0.9999999941	1	1.0000000000	1	0.8000000000	1	0.8650000000	1	0.9530000000	1
11	1	0.9999999997	1	0.9458651665	1	0.8333333333	1	1.0000000000	1	0.8333333321	1	0.8650000000	1	0.9530000000	1
	일치 확률		100%		82%		82%		100%		82%		82%		82%

2,3을 적용한 경우에는 100%의 일치율을 보였으나 그 외의 경우에는 82%의 일치율를 보였다. 이 결과로 미루어 적용요인 1은 적용요인 2나 3에 비하여 상대적으로 더 중요하게 작용함을 알 수 있다. 이는 메소드의 위치를 결정짓는데 대상메소드가 참조하는 필드의 개수보다는 대상메소드가 메소드를 호출하거나 호출되는 횟수가 더 중요하다는 것을 의미한다. 또한, 로지스틱 회귀분석에서 독립변수로 사용되는 적용요인 수가 적은 경우보다 많은 경우에 더 좋은 결과를 보여주고 있다.

여기에서 기존의 리팩토링의 연구에 기초하여 요인을 추출하고 통계기법을 통한 검증이 제대로 이루어진다면, 요인들은 프로그램에서 멤버 메소드 등의 위치를 결정할 수 있는 기준으로 사용할 수 있음을 알 수 있다.

이 연구에서는 리팩토링을 자동화하기 위한 연구를 기반으로 하여 리팩토링 기법 중 하나인 무브 메소드의 적용여부를 판단할 수 있는 세 가지 적용요인을 정의한다. 이렇게 정의된 적용요인들은 로지스틱 회귀분석의 독립변수들로 사용되어 분석되었고, 그 분석결과를 통해 분석에 사용되는 적용요인의 조합에 따라 예측확률이 달라질 수 있음을 알 수 있었다. 또한 분석에 사용되는 적용요인이 많아질수록 정확한 결과를 예측할 수 있었다. 연구 결과로 미루어 매우 흥미로운 사실은 무브 메소드 기법에서 필드의 개수보다는 메소드 참조횟수가 무브 메소드 기법에 영향을 준다는 것이다.

무브 메소드를 적용할 것인가 안할 것인가를 자동으로 결정하는 것은 매우 중요한 일이다. 하지만 이 연구는 자동 적용 결정보다는 메소드들을 대상으로 최적의 위치를 찾아줌으로써 소프트웨어의 안정성을 높이고자 하였다. 메소드의 위치결정을 위한 요인분석에 통계기법을 이용하여 검증함으로써 일반적인 요인으로 사용할 수 있는 척도를 제시했다는 점에서 큰 의미를 가진다.

3.2 필드위치결정을 위한 리팩토링 요인 정의

본 논문에서는 [1]과 같은 방법으로 로지스틱 회귀분석을 통하여 메소드와 같이 객체를 구성하는 중요요소인 필드의 위치를 결정짓기 위한 요인을 추출하였다.

마틴 파울러는 시스템을 개발함에 따라 새로운 클래스가 추가되거나 어떠한 책임을 이리 저리 옮겨야 할 경우가 생긴다고 하였다. 이 경우 기존의 디자인이 적절하지 않을 수 있고, 문제가 발생할 수 있다. 하지만 개발자들은 상황이 바뀌었는데도 어떠한 일도 하지 않으려고 하는 것이다. 라고 지적하고 있다 [8].

무브 메소드를 해야 하는 상황과 비슷하게, 무브 필드도 유사한 상황에 처했을 때 옮길지의 여부에 대한 결정을 내리게 된다. 해당 필드가 자신이 정의된 클래스보다 다른 클래스에서 더 많이 이용되고 있다면, 다른 클래스에 같은 목적의 새로운 필드를 만들고 기존의 필드를 사용하는 모든 부분을 변경한다. 이것이 무브 필드의 전반적인 내용이다.

위에서 언급한 무브 필드의 정의와 같이 고려되었을 때, 이 필드가 다른 클래스에서 set/get 메소드로 이 필드를 간접적으로 많이 사용하고 있다면, 그 메소드를 옮기는 것을 고려해야 한다. 하지만, 그 메소

드가 적당해 보인다면 필드를 옮겨야 한다.

우선 무브 메소드의 경우에는 필드를 참조하는 입장이지만 무브 필드의 경우에는 모두 참조되는 경우만 존재한다. 수동적으로 참조되는 필드의 입장에서 참조의 주체가 되는 메소드는 크게 두 가지로 구분할 수 있다. 참조의 주체가 되는 두 가지 경우의 메소드를 기준으로 하여 무브 필드의 적용요인을 고려하였다. 무브필드의 적용요인은 다음과 같다.

- 적용요인1: get/set 메소드에 의해 참조되는 횟수
- 적용요인2: get/set 메소드를 제외한 일반 메소드에서 참조되는 횟수

4. 결론

본 논문에서는 소프트웨어 안정성을 위한 리팩토링 연구를 기반으로 무브 메소드의 적용요인을 정의하고, 그 요인들을 통계적 기법으로 분석하여 일반적인 요인으로써의 적합여부를 판단하는 처리과정을 제시한 논문[1]을 살펴보았다. 82% 이상의 일치확률을 가지는 이 처리과정에 기초하여 무브필드의 적용요인을 정의하였다. 향후 이 적용요인에 의한 데이터를 획득하여, [1]에서 제시한 방법에 의한 분석을 한다면 본 논문에서 정의한 무브 필드의 적용요인 1,2가 필드의 위치를 결정할 수 있는 요인으로 중요하게 작용할 수 있을 것인지의 여부를 판단할 수 있을 것이다. 또한 여러 리팩토링 기법들 중 메소드나 필드에 관련된 기법들의 요인을 파악하고 다수의 요인을 적용하면 좀 더 정확한 결과가 나올 것이 예상되는 바, 이와 관련된 연구를 진행해 볼만 하다.

참 고 문 헌

[1] 정영애, 박용범, '로지스틱 분석을 이용한 메소드 위치결정방법',정보처리학회 논문지 제 12-D권 제 7호, 2005

[2] 박지훈, '자바 디자인 패턴과 리팩토링', 한빛미디어㈜, 2003

[3] 성웅현, '응용 로지스틱 회귀분석', 도서출판 탐진, 2001

[4] F. Simon, F. Steinbrükner, and C. Lewere- ntz, "Metrics based refactoring", in Proc. European Conf. Software Maintenance and Reengineering IEEE, pp. 30-38, Computer Society, 2001

[5] Katsuhisa Maruyama, Ken-ichi Shima, "Automatic Method refactoring using weighted dependence graphs", Proceedings of the 21st international conference on Software engineering, 1999

[6] Katsuhisa Maruyama, "Automated Method-extraction refactoring by using block-based slicing", Proceedings of the 2001 sympo- sium on Software reusability: putting soft- ware reuse in context, vol 26, pp 236-245, 2001

[7] Kuyul Noh, Changki Kim, Jonggul Park and Jaeha Song, "The Evaluation of Daikon: utilization of Daikon in the POI Data Inspection System", 4WD Team Master of Software Engineering Program School of Computer Science, Carnegie Mellon University, 2002

[8] Martin Fowler, etc, 'Refactoring Improving the Design of Existing Code', Addison Wesley, 1999

[9] Sang-Uk Jeon, "An Approach to Automa- tically Identifying Design Structure for Applying Design Pattern", M.S. thesis of KAIST, 2003

[10] Sang-Uk Jeon, Joon-Sang Lee, and Doo-Hwan Bae, "An Automated Refactoring Approach To Design Pattern-Based Pro- gram Transformations in Java Programs", In Proceedings of 9th Asia-Pacific Software Engineering Conference, Gold Coast in Austrailia, 2002

[11] Stefan Roock , Andreas Havenstein, "Refactoring Tags for automatic refactoring of framework dependent applications", XP2002, 2002

[12] William G. Griswold, 'Program Restruct- uring as an Aid to Software Maintenance PhD Thesis', Dept. of Computer Science & Engineering, University of Washington, 1991

[13] William F. Opdyke and Ralph E.Johnson, Refactoring: An aid in designing application Frameworks and evolving object-oriented systems, In Proceedings of SOOPPA '90: Symposium on Object-Oriented Program- ming Emphasizing Practical Applications. Sep 1990.

[14] W.F. Opdyke, 'Refactoring object-oriented frameworks Ph.D.thesis', Computer Sciences Department, University of Illinois at Urbana Cham-paign, 1992

[15] Yoshio Kataoka, Michael D. Ernst, William G. Griswold, David Notkin, "Automated support for program refactoring using Invariants", Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01), pp. 736-743, 2001